Approximate Membership for Regular Languages modulo the Edit Distance

Antoine Mbaye Ndione, Aurélien Lemay, Joachim Niehren

Approximate Membership for Regular Languages modulo the Edit Distance

Antoine Ndione, Aurélien Lemay, Joachim Niehren

Links *project,* Inria *and* Lifl*, Lille France*

Abstract

We present an efficient probabilistic algorithm for testing approximate membership of words to regular languages modulo the edit distance. Our algorithm runs in polynomial time in the size of the input automaton and the inverse error precision in contrast to all previous approaches, and independently of the size of the input word. We also improve a previous approximate membership tester modulo the Hamming distance such that it runs in polynomial complexity time, but with larger polynomials than for the edit distance.

Keywords: automata, probabilistic algorithms, property testing, tegular word languages.

Introduction

The area of property testing was initiated by Rubinfeld and Sudan in [15] for program checking, and was first applied to combinatorial properties and specifically to graph properties [9, 4]. More recently, it was applied to study properties of hypergraphs [10], boolean functions [2, 13], and geometric functions; see surveys [5, 8, 12]. Approximate membership testing [1, 11, 6] is a special case of property pesting, where one wants to test whether a word belongs to the regular language or to some language in another class.

Approximate membership testing for non-deterministic finite automata (Nfas) is the following problem. Given a word w, a precision value ϵ and an Nfa A, the problem is to discover nonmembership to the language of A if the word w is ϵ-far from it. Intuitively, being ϵ-far from a language means that the correction of an ϵ-fraction of w is insufficient to turn it into a member of the language. Which corrections are permitted depends on the distance notion on words that is chosen. So far, the Hamming distance and the edit distance with moves were considered, but one can also choose the usual edit distance without moves. Since these distances can be ordered decreasingly, any approximate membership tester for the Hamming distance also applies to the edit distance, and any approximate tester for the edit distance can be used for the edit distance with moves. Membership testers whose efficiency does not depend on the size of the input word are most relevant for processing large texts. It means that the tester needs only to inspect a fragment of constant size of any input word when fixing the error precision and the Nfa. In order to provide access to the word's letters, without having to read the word entirely, the tester inputs a reference to an array that contains the word together with its length. In this way, all positions of the word can be drawn from a uniform distribution.

Exact membership testing can be sped up by preprocessing with an approximate membership tester, since whenever the latter returns no, the exact tester can adopt this decision with high probability (or even precisely for one-sided testers) without having to read the entire word. When testing membership for a collection of words, it may thus be sufficient to read only few of them entirely.

precision ε, NFA A over Σ with k strongly connected components, inverse precision $\delta = 1/\varepsilon$, polynomially bounded functions: $p^{i,j,l}(\delta, k, \lvert A\rvert) =_{\text{def}} \delta k^2 \lvert A\rvert^i \log^j(\delta k \lvert A\rvert^l)$		
Distance	Query complexity	Time complexity
Hamming distance Alon et. al. [1] $\rightsquigarrow$ here	$O(p^{2,3,2}(\delta, k, \lvert A\rvert))$	$O(2^{\lvert A\rvert^2} + \delta^k) \rightsquigarrow O(p^{5,3,2}(\delta, k, \lvert A\rvert))$
Edit distance (here)	$O(p^{1,3,1}(\delta, k, \lvert A\rvert))$	$O(p^{2,3,1}(\delta, k, \lvert A\rvert))$
Edit distance with moves Fischer et. al. [6]	$\ln \lvert\Sigma\rvert \ \lvert\Sigma\rvert^{2\delta} \ \delta^4$	$O(\ln \lvert\Sigma\rvert \ \lvert\Sigma\rvert^{2\delta} \ \delta^4 + \lvert A\rvert^{O(\delta)})$

Figure 1: Summary of results on approximate membership testing for regular word languages.

Alon, Krivelevich and Newmann [1] showed that approximate membership testing with constant query complexity is indeed possible for regular languages modulo the Hamming distance, for fixed automata and error precision. As argued above, their algorithm can equally be used to test approximate membership for larger distances, such as the edit distance or the edit distance with moves. The edit distance with move is the natural distance notion obtained if considering words as directed graphs with deletion and addition of edges but may be too relaxed for some applications. Fischer, Magniez, and de Rougemont [6] proposed another approximation algorithm for regular languages modulo the edit distance with moves, which may be exponential in the inverse of the error precision though.

The state of the art leaves two main problems open. First, all existing testers may require exponential time in the size of the input automaton or the inverse error precision. It is unknown whether polynomial time algorithms exist. Second, input automata are assumed to be deterministic, which may induce another exponential blow up for determinization. The question is whether there exist an algorithm that can also deal with nondeterministic automata in polynomial time. For these two reasons, the previous algorithms fail to be efficient which limits their potential for use in practice.

Our first contribution is an approximate membership tester for NFAs modulo the Hamming distance, which runs in polynomial time depending on the size of the input NFA and the inverse error precision, independently of the size of the input word. The new algorithm is obtained by reformulating Alon, Krivelevich and Newmann's [1] algorithm based on the notion of infeasible fragments of words that we introduce. In order to decide feasability, we have to decide k-step reachability of NFA states in polynomial time, but independently of k. As we show, this is indeed possible under the assumption that addition and multiplication on natural numbers can be done in constant time. Here we apply ideas from algorithms for computing the Chrobak normal form of NFAs over a single letter alphabet [7].

As our second contribution, we show that approximate membership for NFAs modulo the edit distance can even be tested more efficiently. Our new tester is based on the notion of blocking fragments that we introduce. In order to decide whether a fragment is blocking, we have to decide reachability for NFA states. The reason for which we can relax k-reachability as for the Hamming distance is that errors can be corrected by letter insertion for the edit distance. Reachability can be decided in linear time in the size of the NFA, and thus more efficiently than k-step reachability. Therefore, the degree of the polynomial for the complexity of approximate membership testing can be reduced by 3. We summarize the new state of the art including our results in Figure 1.

Outline. In Section 1 we start with preliminaries on finite automata and approximate membership testing. Section 2 introduces informally the notions of infeasible and blocking fragments of words and illustrates their relevance for approximate membership testing by example. Section 3 presents the formal definitions of blocking and infeasible fragments and presents polynomial time algorithms to decide these properties. The next two sections present our approximate membership tester for NFAs modulo the edit distance. In Section 4, we treat the case of strongly connected NFAs based on the notion of blocking intervals, and in Section 5, we generalize this algorithm to arbitrary NFAs with multiple strongly connected components, while relying on blocking fragments. In Section 6, we sketch an analogous approximate membership tester for NFAs modulo the Hamming distance, in which the notion of feasible fragments becomes essential.

1. Preliminaries

We recall preliminaries on finite automata and property testing. In particular, we introduce the notion of a fragment of a word and show how to run a finite automaton on a fragment. We also recall various distances between words as well as the notion of approximate membership testing.

1.1. Words, Fragments, and Finite Automata

Let $\mathbb{N}$ be the set of non-zero natural numbers, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, and $\mathbb{R}_{\geq 0}$ the set of positive real numbers. Given a finite set S we write $|S| \in \mathbb{N}_0$ for its cardinality. An *alphabet* Σ is a finite set. We write Σ^* for the set of words with letters in Σ, that is $\Sigma^* = \uplus_{n \in \mathbb{N}_0} \Sigma^n$. We denote by $a_1 \dots a_n$ the word $(a_1, \dots, a_n) \in \Sigma^n$ where $n \geq 0$. The empty word is the unique element of Σ^0. The length of $w = a_1 \dots a_n$ is $|w| = n$, the set of positions $pos(w) = \{1, \dots, n\}$, and the domain $dom(w) = pos(w) \cup \{0\}$. Note that the domain of the empty word is non-empty. We denote the concatenation of two words w and w' by $w \cdot w'$. In particular, note that $a \cdot w$ is the concatenation of the one-letter word $a \in \Sigma^1$ with the word w.

A fragment of a word w is a subset of positions $F \subseteq pos(w)$. We define the domain of a fragment F by $dom(F) = \{0\} \cup \{i-1, i \mid i \in F\}$. Hence $dom(pos(w)) = dom(w)$ even for the empty word. An (half-open) interval of w is a fragment $I =]i, j]$ of w without holes, that is the set $\{i+1, \dots, j\}$ where $i < j$ are positions of w. A factor of a word w is another word w' such that $w = w_1 \cdot w' \cdot w_2$ for some w_1 and w_2. If $w = a_1 \dots a_n$ and $]i, j]$ is an interval of w then we denote the factor of w at $]i, j]$ by $w]i, j] = a_{i+1} \dots a_j$.

A non-deterministic finite automaton (NFA) is a tuple $A = (\Sigma, Q, \Delta, init, fin)$, where Σ is an alphabet of letters, Q a finite set of states, $\Delta \subseteq Q \times \Sigma \times Q$ a transition relation, and $init, fin \subseteq Q$ the subsets of initial and final states. If $(q, a, q') \in \Delta$, we say that $q \xrightarrow{a} q'$ is a transition or rule of A. Furthermore, we write $\xrightarrow{a}_A$ for the relation $\{(q, q') \mid (q, a, q') \in \Delta\}$ and $\rightarrow_A$ for the one-step reachability relation $\cup_{a \in \Sigma} \xrightarrow{a}_A$. The k-reachability relation is defined inductively by $\rightarrow_A^k = \rightarrow_A^{k-1} \circ \rightarrow_A$ and $\rightarrow_A^0 = \{(q, q) \mid q \in Q\}$. The reachability relation of A is the union of all k-reachability relations $\rightarrow_A^* = \cup_{k \geq 0} \rightarrow_A^k$.

A *quasi-run* of an NFA A on a fragment F of a word $w = a_1 \dots a_n$ is a function $r : dom(F) \to Q$ such that $r(i-1) \xrightarrow{a_i} r(i)$ is a transition of A for all $i \in F$. A quasi-run is called total if its domain is $dom(w)$. Note that $q \rightarrow_A^* q'$ if and only if there exists a word $w \in \Sigma^n$ and a total quasi-run r on w by A such that $r(0) = q$ and $r(n) = q'$.

A run r of A on a fragment F of w is a quasi-run of A on this fragment such that $r(0) \in init$. A run is called successful if it is total and satisfies $r(n) \in fin$. As usual, we say that w is recognized by A if there exists a successful run of A on w. The language $L(A) \subseteq \Sigma^*$ is the set of all words w recognized by A. We call a language $L \subseteq \Sigma^*$ regular if it is equal to $L(A)$ for some NFA A. The size of an NFA A is the sum of the number of its states and rules $|A| = |Q| + |\Delta|$.

1.2. Distance Notions and Property Testing

A distance on words with alphabet Σ is a binary real-valued function $d \subseteq \Sigma^* \times \Sigma^* \to \mathbb{R}^{\infty}_{\geq 0}$ such that $d(w,w) = 0$ and $d(w,w') + d(w',w'') \geq d(w,w'')$. The distance between a word and a non-empty language is $d(w,L) = \inf\{d(w,w') \mid w' \in L\}$. A word w of length n is called ε-far from a non-empty language L with respect to the distance if $d(w,L) > \varepsilon n$, and ε-close otherwise. Note that ϵ-farness and ϵ-closeness are defined with respect to the relative distance normalized by the length of the word (and not with respect to the absolute distance).

The Hamming distance $d_{hamming}(w,w')$ between two words $w = a_1 \dots a_n$ and $w' = a'_1 \dots a'_m$ is the least number of letter-exchange operations by which the two words become equal. For words of the same length $m = n$, this is the number of positions i such that $a_i \neq a'_i$, and for words of different length, this is ∞. The edit distance $d_{edit}(w,w')$ between two words w and w' is the least number of insertion, deletion, and letter-exchange operations needed to transform w into w'. For instance, $d_{hamming}(001100, 110011) = 6$ and $d_{edit}(001100, 110011) = 4$ since we can insert 11 at the beginning of the left word and delete 00 at its end. Clearly the edit distance between w and w' is always smaller or equal than their Hamming distance.

We recall the problem of approximate membership testing to regular languages defined by NFAs modulo some distance.

Definition 1. *An approximate membership tester for* NFA*s A with alphabet Σ and a distance function $d : \Sigma^* \times \Sigma^* \to \mathbb{R}^{\infty}_{\geq 0}$ is a terminating probabilistic algorithm that reads as input a precision value $\varepsilon \in \mathbb{R}_{>0}$, a natural number $n \in \mathbb{N}_0$, and a reference r_w to an array containing a word $w \in \Sigma^n$ and outputs* CLOSE *or* NO *such that:*

- *if $w \in L(A)$ then with probability $\frac{2}{3}$ it outputs* CLOSE*, and*
- *if $d(w, L(A)) > \varepsilon n$ then with probability $\frac{2}{3}$ the output is* NO*.*

When ε-close non-members of the language are received as input, approximate membership testers are allowed to answer both CLOSE or NO without any particular requirements. Given a tester M, we can obtain another tester M' with higher probability than 2/3 by repeating M sufficiently often. Also note that all testers presented in the present paper will be one-sided in that they will always answer CLOSE for all words of the language. Hence, NO-answers of one-sided testers are always correct (not only with high probability).

Since the edit distance between w and w' is always smaller or equal than their Hamming distance, any approximate membership tester modulo the Hamming distance will also be an approximate membership tester modulo the edit distance.

The *query complexity* of a tester is the maximal number of positions of the input word that it may access until termination. We are particularly interested in testers with constant query complexity in the size of the input word, so that the number of read operations may only depend on the NFA and the error precision. The *time complexity* of a tester is the maximal number of computation steps it might require until termination. Here, we assume that all arithmetic operations, by which to access the letters of the word in the array, take time in $O(1)$ rather than in $O(\log n)$.

2. Examples

We illustrate informally how to witness farness with respect to the Hamming distance or to the edit distance by large collections of small "infeasible" or respectively "blocking" fragments. The existence of such witnesses can then be tested by probabilistic algorithms. The precise definitions will be given in the next section.

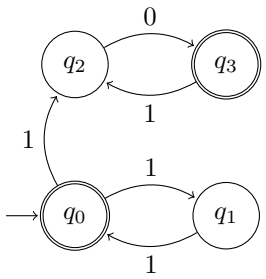

Figure 2: An NFA for $L = (11)^*(10)^*$.

We consider the regular language $L = (11)^*(10)^*$ over the alphabet $\Sigma = \{0, 1\}$. This language is recognized by the NFA with 4 states in Figure 2. This NFA has $k = 2$ strongly connected components, namely $\{q_0, q_1\}$ and $\{q_2, q_3\}$ which correspond to the two Kleene star operators in the definition of L.

2.1. Intervals

We consider words $u_m = (01)^m$ with $m > 0$ which do not belong to L. They can be corrected, however, to become a member of L by flipping all 0s to 1. It is apparent that corrections with fewer relabeling operations do not exist, so the Hamming distance of u_m to L is equal to m. In other words, u_m is at least $1/2$-far from L with respect to the Hamming distance. With only 2 insertion operations, however, we can correct u_m so that it belongs to L, by adding a 1 in front and a 0 at the end of u_m. Therefore the edit distance of u_m to L is at most 2, so u_m is $1/m$-close to L for the edit distance.

We next present a collection of intervals that witness for the Hamming distance that u_m is far from L. We consider the intervals $]i, i + 2]$ of u_m where $0 \leq i < 2m - 1$ and i is even. These intervals are infeasible for the NFA in the following sense: First note that the factor of u_m at interval $]i, i + 2]$ is equal to 01. Words of even length can reach states q_0 and q_3 only. When proceeding from there with word 01 all possible runs of the NFA get stuck. This shows that no word with factor 01 at any even position may belong to the language. Hence, at least 50% of all intervals of length 2 are infeasible for this NFA. An analogous argument for odd positions shows that all of them are infeasible (but a fixed percentage will be enough for our algorithm).

When it comes to farness for the edit distance, the precise position of an interval does not matter but only the factor that it defines. The reason is that new letters can be inserted for correction before or after the factor. Therefore, the notion of infeasible intervals can be weakened to the notion of blocking intervals. These are intervals that do permit to run the NFA under consideration. Indeed, none of the intervals $]i, i + 2]$ of u_m is blocking in this sense. If i is odd, then the NFA can be run on the i-factor 10 from states q_0 or q_3 – even though no word of odd length can reach these states – and if i is even, the NFA can be run on the i-factor 01 from states q_1 and q_2 – even though no word of even length will end up there.

2.2. Fragments

We next show that collections of intervals are not always sufficient to witness farness. In order to see this, we consider the words $v_m = (10)^m(11)^m$ with $m > 0$. Again, no v_m belongs to L but all of them can be corrected by flipping 0s to 1. There is no better way even not if permitting insertion and deletion operations. Therefore, v_m is $1/4$-far from L both with respect to the edit and the Hamming distance. Notice that $]2m - 1, 2m + 1]$ is the only blocking or

infeasible interval of v_m, hence most of the 'small' collections of intervals of length 2 doesn't witness 1/4-farness of v_m.

We now consider the fragments $]i, i+2] \cup]j, j+2]$ where $0 \leq i < 2m-1 \leq j < 4m-1$. The pair of factors of v_m at these fragments are all equal to $(10, 11)$, and thus blocked in the following sense: Before reading the first factor arbitrary states are reachable, but when continuing with the first factor 10, only state q_3 can be reached. From there, we can reach states q_2 and q_3 over arbitrary words, given that we do not make any assumptions on the word between the two factors (even not on its length). From there, if we try to continue with the second factor 11, we get blocked in state q_2. This shows that all above fragments are blocking, and they constitute roughly 1/4th of all fragments with 2 intervals of length 2.

2.3. Sampling Algorithms

All our algorithms will be based on probabilistic sampling for testing whether a large fraction of "short" fragments is blocking or infeasible. Most typically, we might want to test whether a word contains an ε-fraction of blocking fragments of some fixed length, say 2 for instance. This can be tested by a probabilistic algorithm as follows: parametrized by a positive real c, it draws randomly c/ε fragments of length 2 of the input word from a uniform distribution, and answers NO if at least one of them is blocking, and YES otherwise. This algorithm will detect some blocking fragment with probability at least $1-(1-\varepsilon)^{c/\varepsilon} \geq 1- \ e^c \xrightarrow{c\to\infty} 1$. Therefore, it answers correctly with NO with high probability, for instance, with probability 0.9999 if we choose a parameter $c \geq \ln(0.0001)$.

This kind of sampling algorithm can be used with $\varepsilon = 1/4$, for instance, in order to show that v_m is far from L modulo the edit distance. It can also be applied to show that u_m is far from L modulo the Hamming distance, by considering infeasible intervals of size 2 and choosing $\varepsilon = 1/2$. For testing membership of other words to L, one might expect that c increases with the distance from L. How to choose c for given NFA A and error precision ϵ, is less obvious though.

3. Blocking and Infeasible Fragments

We next define the notions of blocking and infeasible fragments formally and show how to decide these properties in polynomial time in the size of the fragment and the NFA, but independently of the size of the word.

Since intervals are particular fragments, we also reintroduce infeasible intervals, which were called "infeasible runs" in [1]. All other notions are original to the present article.

For all what follows, we fix an NFA $A = (\Sigma, Q, \Delta, \mathit{init}, \mathit{fin})$, a length $n \in \mathbb{N}_0$ and a word $w \in \Sigma^n$.

Definition 2. *A run r of an* NFA *A on a fragment F of w (of size n) is* blocking *if:*

1. *there exist elements $i, j \in dom(F)$ with $i < j$ such that $r(i) \not\to_A^* r(j)$ – in this case we can choose i, j such that $]i, j]$ is disjoint from F –, or*
2. *the maximal element m of $dom(r)$ satisfies $r(m) \not\to_A^* \mathit{fin}$, or $m = n$ and $r(m) \notin \mathit{fin}$.*

We call a fragment F of w blocking *for A if all runs of A on F are blocking.*

Since intervals are fragments without holes, this definition introduces the notion of blocking intervals as a special case. Furthermore, note that no fragment of a word in $L(A)$ is blocking.

Proposition 3. *Whether a fragment F of a word w is blocking for an* NFA *A can be decided in time $O(|F||A|)$ by an algorithm that receives as inputs the* NFA *A and a reference to an array containing the word w, and the length of w.*

Note that the whole word w cannot be read in time $O(|F||A|)$. Therefore, a reference to an array is passed as input that contains the word, and in addition, the length of this word.

Proof. We define a non-deterministic evaluator for A on fragments F of the word. It reads the positions of F in increasing order, applies automata transitions between subsequent positions, and whenever meeting a hole of F then it jumps to all states that are reachable from the current state set P. This set is denoted by $\rightarrow_A^*(P)$. For all words $w = a_1 \dots a_n$, elements $i \in pos(w)$, non-empty fragments $F \subseteq \{i, \dots, n\}$, and state sets $P \subseteq Q$ we define:

$$eval_A(F) = eval'_A(0, init, F)$$
$$eval'_A(i-1, P, F) = \begin{cases} eval'_A(i, \xrightarrow{a_i}_A (P), F \setminus \{i\}) & \text{if } i = \min(F) \\ eval'_A(j-1, \rightarrow_A^* (P), F) & \text{if } j = \min(F) > i \end{cases}$$
$$eval'_A(i, P, \emptyset) = \begin{cases} P & \text{if } i = n \\ \rightarrow_A^* (P) & \text{if } i < n \end{cases}$$

Note that $eval_A(F) \cap fin = \emptyset$ if and only if F is blocking for A. Furthermore, $eval_A(F)$ can be computed in time $O(|F||A|)$ by an algorithm that receives as inputs a fragment F, a reference to an array containing w, and the length n of w. This can be done by computing the least fixed point of a ground Datalog program of size $O(|F||A|)$, and thus in this time (see e.g. [**?**]). Note that for all P the set of reachable states $\rightarrow_A^* (P)$ can be computed in time $O(|A|)$. Note also that the computation time is independent of the length of w. □

Definition 4. *A run r of an* NFA *A on a fragment F of w (of length n) is called* infeasible *if:*

1. *there exist elements $i, j \in dom(F)$ such that $i < j$ and $r(i) \not\rightarrow_A^{j-i} r(j)$ – in this case, we can choose i, j such that $]i, j]$ is disjoint from F –, or*
2. *the maximal element m of $dom(r)$ satisfies $r(m) \not\rightarrow_A^{n-m} fin$.*

We call a fragment F of w infeasible *for A if all runs of A on F are infeasible.*

Note that blocking runs are infeasible, and thus blocking fragments are infeasible too. Furthermore, no successful run is infeasible, so no word in $L(A)$ may be infeasible nor blocking.

The following proposition for NFAs will help us get rid of the exponential dependency on the automaton size in Alon et. al.'s algorithm.

Proposition 5. *Whether a fragment F of a word w is infeasible for an* NFA *A can be decided in time $O(|F||A|^3 + |A|^5)$, when receiving as input a reference to an array containing w and its length (so that the whole word does not need to be read).*

Proof. The decision procedure for infeasibility of fragments is similar to deciding whether a fragment is blocking, except that holes in fragments need to be evaluated more strictly. Therefore, we define a strict evaluator for fragments which behaves like the previous evaluator, except that it respect the number of missing positions in holes of fragments. For any word $w = a_1 \dots a_n$, element $i \in pos(w)$, non-empty fragment $F \subseteq \{i, \dots, n\}$, and state set $P \subseteq Q$ we define:

$$s\_eval_A(F) = s\_eval'_A(0, init, F)$$
$$s\_eval'_A(i-1, P, F) = \begin{cases} s\_eval'_A(i, \xrightarrow{a_i}_A (P), F \setminus \{i\}) & \text{if } i = \min(F) \\ s\_eval'_A(j-1, \rightarrow_A^{j-i} (P), F) & \text{if } j = \min(F) > i \end{cases}$$
$$s\_eval'_A(i, P, \emptyset) = \rightarrow_A^{n-i} (P)$$

Note that $s\_eval_A(F) \cap fin = \emptyset$ if and only if fragment F of w is infeasible for A. We can now test whether fragment F of w is infeasible for A by computing $s\_eval_A(F)$ and checking

whether $s\_eval_A(F) \cap fin = \emptyset$. The following Lemma 6 implies that $s\_eval_A(F)$ can be computed recursively along its definition, with $O(|F|)$ recursive calls each of which costs $O(|A|^3)$, after a preprocessing time of $O(|A|^5)$. So the overall computation time is in $O(|F||A|^3 + |A|^5)$ as stated by the proposition. □

Lemma 6. *For any* NFA A *we can compute in preprocessing time* $O(|A|^5)$ *an algorithm that receives as inputs a subset* P *of states of* A *and a natural number* $m \in \mathbb{N}_0$ *and computes in time* $O(|A|^3)$ *the set* $\rightarrow_A^m (P)$*, so in time independent of* m*.*

Proof. For any pair of states $(q, q') \in Q$, we compute an NFA $A_q^{q'}$ with a single letter alphabet $(\{0\}, Q, \Delta_A^0, \{q\}, \{q'\})$ where $\Delta_A^0 = \{(q_1, 0, q_2) \mid q_1 \rightarrow_A q_2\}$. We then convert all $A_q^{q'}$ into their Chrobak normal form by using the algorithm in [7]. This takes time $O(|Q|^3)$ for each pair (p, p') and thus $O(|A|^5)$ all together. Recall that a Chrobak normal form of a single-letter NFA $A_q^{q'}$ is a single-letter NFA $B_q^{q'}$ that recognizes the same language, such that the digraph of $B_q^{q'}$ consists of a single path with at most $|Q|^2$ states, succeeded by a non-deterministic choice of a set of disjoint cycles whose total sizes is at most $|Q|$. One can thus precompute in time $O(|Q|^2)$ the length of the path and an array containing its states, and in time $O(|Q|)$ the length of each cycle and an array containing its states.

We show next – once having precomputed the sizes of the path and the cycles – that given a pair of states $(q, q') \in Q^2$ and $m \in \mathbb{N}_0$, we can check in time $O(|Q|)$ whether $q \rightarrow_A^m q'$. This property is equivalent to that $B_q^{q'}$ accepts some word of length m. If m is smaller than the length of the path of $B_q^{q'}$, this can be done by selecting the m'th state of the path in its array, and testing whether it is final for $B_q^{q'}$. Otherwise, we compute $l' = m - l$ where l is the length of the path, and for all cycles of $B_q^{q'}$ the state that is reached with l' steps. This can be done by computing the remainder of l' by division modulo the length of the cycle, and accessing the state of this remainder in the array of the cycle. It then remains to check whether any of the computed states is final. For each cycle, all these operations can be done in $O(1)$. Since the number of cycles is in $O(|Q|)$ the overall time is in $O(|Q|)$ too.

In order to compute $\rightarrow_A^m (P)$ for some m and P, we check whether $q \rightarrow_A^m q'$ for all pairs $(q, q') \in P \times Q$. This takes $|Q|^2$ time $O(|Q|)$, and thus can be done in time $O(|A|^3)$. □

4. Membership for Strongly Connected NFAs modulo the Edit Distance

We present an approximate membership tester with respect to the edit distance for regular languages defined by NFAs that are strongly connected. These are NFAs such that $q \rightarrow_A^* q'$ for any two states q and q'.

Let $\varepsilon > 0$, $\delta = 1/\varepsilon$ and $A = (\Sigma, Q, \Delta, init, fin)$ a strongly connected NFA containing some initial and some final state. Since A is strongly connected with initial and final states, then $L(A)$ is not empty. Clearly, there exists a run of A on any fragment of any word in $L(A)$, that is, no fragment or interval of any word in the language is blocking. In contrast, words that are ε-far from the language with respect to the edit distance must have many short blocking intervals:

Lemma 7. *Let* $\gamma = 4\delta(|Q| + 1)$*,* $n \geq 8\gamma\lceil\log(\gamma)\rceil$ *a natural number, and* $w \in \Sigma^*$ *a word of size at most* n*, and* d *the edit distance from* w *to* $L(A)$*. If* $d > \varepsilon n$ *then there exists a power of two* $l = 2^i$ *in* $[2, \gamma]$ *such that the number of intervals of length* $2l$ *that are blocking for* A *is at least* $n\beta_l$ *where* $\beta_l = l/(2\gamma\lceil\log(\gamma)\rceil)$*.*

The assumptions of the lemma imply $|Q| \leq \epsilon n \leq |w| \leq n$, since $|Q| + |w| \geq \max(|Q|, |w|) \geq d$ holds generally for the edit distance for $L(A) \neq \emptyset$, and $|w| \geq d - |Q| \geq n\epsilon - |Q| \geq |Q|$ by

these assumptions, so that $max(|w|, |Q|) = |w|$. In the application in this section, we will choose $|w| = n$, but for the general case in Section 5, the lemma will be applied to some large interval of a word of size n. Note also that the number of blocking intervals increases with l. Furthermore, the lower bounds β_l of the linear growth rate increases monotonically with l such that for all $l \leq \gamma$:

$$\beta_l \leq \beta_\gamma = 1/(2\log(\gamma)) \leq 1/4$$

Proof. Let w be a word of size at most $n \geq \gamma\lceil\log(\gamma)\rceil$ whose edit distance from $L(A)$ is at least $d > \varepsilon n$. We consider the unique decomposition $0 = i_0 < \ldots < i_h = |w|$ such that all $I_j =]i_{j-1}, i_j[$ are maximal non-blocking intervals, where $1 \leq j \leq h$. Since A is strongly connected, we can repair w such that it becomes a word of $L(A)$ by first deleting the letters of w at positions i_j and then inserting words of length at most $|Q|$ after all i_j where $0 \leq j \leq h$. Without lost of generality, and to ease up the repair strategy presentation, we show this only in the case where all intervals I_j are not empty. Note that I_j is empty only if the letter at position i_j does not appear in any word of $L(A)$. Then let r_j be some run on interval I_j of w. The word inserted at i_0 is chosen such that it has a total run from *init* to $r_1(i_0)$; this is possible since A is strongly connected and has an initial state. For all $1 \leq j < h$, the word inserted after i_j is chosen such that it has a total quasi-run from $r_j(i_j)$ to $r_{j+1}(i_j)$ by A. The word inserted after i_h must have a total quasi-run from $r_h(i_h)$ to *fin* by A'. This is possible since we assume that *fin* is non-empty. Clearly, the repaired word belongs to $L(A)$. The overall correction costs in terms of letter insertion and deletion operations is $h + 1 + |Q|(h + 1)$, so that $d \leq h + 1 + |Q|(h + 1)$. Hence $\varepsilon n \leq h + 1 + |Q|(h + 1)$ so that:

$$h \geq \frac{4n}{\gamma} - 1 \geq \frac{3n}{\gamma}$$

The last inequation follows from $n/\gamma \geq 2$, which in turn is a consequence of $n \geq \gamma\log(\gamma)$ and $\gamma \geq 4$ (so that $\log\gamma \geq 2$).

Let $S = \{I_j \mid 0 \leq j \leq h - 1\}$. We call an interval of w small if its size is at most γ and big otherwise. We next estimate the numbers of small intervals I in S such that I is blocking for A. For all $2 \leq l \leq 2\gamma$, let S_l be the subset of S of small intervals whose size belongs to $]l/2, \min(l, \gamma)]$. The number of big intervals of w is at most $|w|/\gamma \leq n/\gamma$, so that the number of small intervals in S is at least $h - n/\gamma$. Since $\cup_{i=1}^{\lceil\log(\gamma)\rceil} S_{2^i}$ is a partition of the set of small intervals of S, the above lower bound for h implies:

$$\sum_{i=1}^{\lceil\log(\gamma)\rceil} |S_{2^i}| \geq 2n/\gamma$$

Therefore, there exists a number $l = 2^i$ with $2 \leq l \leq \gamma$ and $|S_l| \geq 2n/\gamma\lceil\log(\gamma)\rceil$. We fix such an index l.

We are now interested in the cardinality of the set W_{2l}, which contains all intervals of w of size in $]l, 2l]$ that are blocking for A. We next estimate the cardinality of W_{2l}. Let i_1, i_2, i_3, i_4 be the two smallest and the two greatest indexes in $\{j \mid I_j \in S_l\}$ respectively. Every interval $I_j \in S_l$ where $j \notin \{i_1, i_2, i_3, i_4\}$ is subsumed by $[l, n - l]$ and thus belongs to l intervals of W_{2l}. Conversely, every interval of W_{2l} may contain at most 3 intervals from S_l, since the latter are non-overlapping and of size at least $1 + l/2$. Hence:

$$\begin{array}{rcll} |W_{2l}| & \geq & \frac{l\ (|S_l|-4)}{3} & \\ & \geq & \frac{l}{3}(\frac{2n}{\gamma\lceil\log(\gamma)\rceil} - 4) & \text{since } |S_l| \geq 2n/\gamma\lceil\log(\gamma)\rceil \\ & \geq & \frac{l}{3}(\frac{2n}{\gamma\lceil\log(\gamma)\rceil} - \frac{n}{2\gamma\lceil\log(\gamma)\rceil}) & \text{since } n \geq 8\gamma\lceil\log(\gamma)\rceil \\ & = & \frac{nl}{2\gamma\lceil\log(\gamma)\rceil} = n\beta_l & \end{array}$$

□

```
fun membership_edit_connect_A(ε, n, r_w)
  // r_w reference to an array containing some word w ∈ Σ^n
  // 0 < ε < 1 precision value
  // A = (Σ, Q, Δ, init, fin) strongly connected NFA with init ≠ ∅ and fin ≠ ∅
  let δ = 1/ε          // inverse precision
  let γ = 4δ(|Q| + 1) // size bound for small intervals
  if n < 8γ⌈log(γ)⌉ then
     if w ∈ L(A) then return CLOSE else return NO
     // membership for small words w can be decided by running NFA A on w;
     // this needs time  O(γlog(γ) |Q|)
  else // word w is sufficiently long
     for i = 1 to ⌈log(γ)⌉ do
       let l = 2^i
       let β_l = l/2γ⌈log(γ)⌉ // fraction of blocking intervals of size 2l by Lemma 7
       for j_1 = 1 to 2/β_l
         select j ∈ [0, n − 2l] uniformly
         if interval ]j, j + 2l] of w is blocking wrt. A
            then return NO and exit else skip
            // this can be tested in time O(2l|Q|) by Proposition 3
       end
     end // no small blocking interval of w found
     return CLOSE
```

Figure 3: An approximate membership tester for strongly connected NFAs with respect to the edit distance.

The above lemma tells us that we can test approximate membership for strongly connected NFAs with respect to the edit distance by selecting sufficiently many small intervals randomly and testing whether they are all non-blocking, and it provides estimations for the sizes and numbers of small intervals that needed to be considered. However, we cannot know the precise size $l = 2^i$ of small intervals, so we have to try out all possible sizes.

Proposition 8. *Algorithm membership_edit_connect$_A$ in Figure 3 is a one-sided approximate membership tester for words w in regular languages recognized by strongly connected* NFA*s A with respect to the edit distance. The query complexity is in $O(\delta|A|\log^2(\delta|A|))$ and the time complexity in $O(\delta|A|^2\log^2(\delta|A|))$, where $\delta = 1/\epsilon$ is the inverse precision value.*

Note that the upper bound for the time complexity $O(p^{2,2,1}(\delta, k, |A|))$ where $k = 1$ is slightly better than $O(p^{2,3,1}(\delta, k, |A|))$ as we promised in the introduction for the general case, and similarly for the query complexity.

Proof. Algorithm *membership_edit_connect$_A(\varepsilon, n, r)$*, where r is a reference to an array containing a word w of size n, first checks whether word w is sufficiently long to apply Lemma 7, that is whether $n \geq 8\gamma\lceil\log(\gamma)\rceil$ with $\gamma = 4\delta(|Q| + 1)$. This can be done in time $O(|A|)$ without traversing the word, since its size n is passed as input. Furthermore, note that we assume that arithmetic operations can be done in size $O(1)$.

The algorithm always returns CLOSE if $w \in L(A)$, since in this case no interval of w is blocking. This shows that the algorithm is one-sided. We next assume that $d(w, L(A)) \geq \varepsilon n$ and want to compute the probability that the algorithm answers NO. By Lemma 7 there exists a power of two $l \in [2, \gamma]$ such that the number of intervals of length $2l$ that are blocking for A is at least $n\beta_l$ where $\beta_l = l/2\gamma\lceil\log(\gamma)\rceil$. Our algorithm misses all these intervals with probability:

$$(1 - \beta_l)^{2/\beta_l} = (1 - 2/(2/\beta_l))^{2/\beta_l} \leq e^{-2} < 1/3.$$

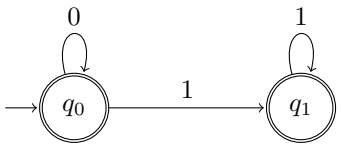

Figure 4: An NFA recognizing language 0^*1^*.

The algorithm will thus answer NO correctly with probability of at least 2/3. The query complexity of the algorithm for sufficiently long words is in $O(\sum_{\{l \mid l=2^i,\ 1\leq i\leq \lceil \log(\gamma)\rceil\}} \sum_{j=1}^{2/\beta_l} 2l)$ and thus in $O(\delta|A|\log^2(\delta|A|))$. The time complexity is by a factor of $|A|$ higher, since NFA A is to be run on all selected intervals of w in order to check whether they are non-blocking (Proposition 3) and thus it is in $O(\delta|A|^2\log^2(\delta|A|))$. □

5. Membership for General NFAs modulo the Edit Distance

We next treat approximate membership for general NFAs with respect to the edit distance. This problem is more difficult than the case of connected NFAs, since the correction algorithm in the proof of Lemma 7 fails. Indeed, this lemma fails for general NFAs, so what we need is a proper generalization.

For illustration, we consider the regular language $L = 0^*1^*$ which can be recognized by the NFA with 2 states in Figure 4. This NFA has $k = 2$ strongly connected components. We consider the collection of words $w_m = 1^m0^m$ with $m \in \mathbb{N}$. A word w_m has length $n = 2m$ and edit distance m from L, so it is 1/2-far from L. An interval I of w_m is blocking for the above NFA if and only if the factor of w at interval I subsumes 10, that is, if $\{m, m+1\} \subseteq I$. Thus, for all $1 \leq l \leq m$ the number of blocking intervals of w_m of size l is equal to $l-1$. This number is too small, in that it fails to grow linearly with n for any l in contrast to what Lemma 7 would predict for strongly connected NFAs. Similar to the example in Section 2.2, this problem can be solved by looking into fragments F with one hole. These are unions of two disjoint subsequent intervals I_1 and I_2 such that $F = I_1 \cup I_2$. Such fragments F are blocking for A if the factor of w_i at I_1 contains the letter 1 while the factor of w_i at I_2 contains the letter 0. Therefore, the number of blocking fragments of size $|F| = 2$ is m^2. This number grows linearly with the total number of fragments which are a union of two intervals ($\leq 4m^2$), so we can detect farness from 0^*1^* by inspecting sufficiently many fragments with 2 positions, which need not to be subsequent.

Let $A = (\Sigma, Q, \Delta, \mathit{init}, \mathit{fin})$ be an NFA. Without loss of generality, we assume that A is productive, i.e. that every state in Q is reachable from *init* and co-reachable from *fin*. A connected component of A is a maximal subset of states of A that is strongly connected. Note that the strongly connected components of A partition Q. Let k be the number of strongly connected components of A. We also fix a precision value $\epsilon > 0$ and its inverse $\delta = 1/\varepsilon$.

Definition 9. *A* component path of A *is a sequence* $\Pi = (Q_1, \ldots, Q_j)$ *of pairwise distinct strongly connected components of* A *such that for all* $1 < h \leq j$ *some state of* Q_h *is reachable from some state of* Q_{h-1}.

In this case, all states of later components of Π are reachable from all states of earlier components. Furthermore, note that the length of any component path is at most k, that is $0 \leq j \leq k$.

Let $Q' \subseteq Q$ be a subset of states of A. The restriction of A to Q' is the NFA $A(Q') = (\Sigma, Q', \Delta', \mathit{init}', \mathit{fin}')$ with state set Q', initial states $\mathit{init}' = Q'$, final states $\mathit{fin}' = Q'$, and

transition relation $\Delta' = \Delta \cap (Q' \times \Sigma \times Q')$. The restriction of A to a component path Π is the automaton $A(\Pi) = A(Q_1 \cup \ldots \cup Q_j)$.

A decomposition of word w of length n along a component path $\Pi = (Q_1, \ldots, Q_j)$ is a sequence of integers $J = (i_0, i_1, \ldots, i_j)$ such that $0 = i_0 < \ldots < i_j = n$. Notice that any decomposition depends on the length of the word and the length of the component path (which we sometimes leave implicit in the context).

Definition 10. *A word $w \in \Sigma^n$ is ε-far from a path $(Q_1, \ldots, Q_j)$ of A if for all decompositions $0 = i_0 < \ldots < i_j = n$ there exists some integer h such that:*

$$d_{edit}(w]i_{h-1}, i_h], L(A(Q_h)) > \varepsilon n$$

A word is called ε-close from Π if it is not ε-far from it. The next lemma shows that sufficiently long words far from $L(A)$ are also far from any component path of A.

Lemma 11. *Let $\alpha = 2(k+1)|Q|\delta$ where k is the number of strongly connected components of A and let $w \in \Sigma^n$ be a word of length $n \geq \alpha$. If word w is ε-far from $L(A)$ then it is $\frac{\varepsilon}{2k}$-far from any component path of A.*

Proof. Let a word $w \in \Sigma^n$ with $n \geq \alpha$ be $\frac{\epsilon}{2k}$-close to some component path $\Pi = (Q_1, \ldots, Q_j)$ of A. We will then show that w is ϵ-close to $L(A)$. By assumption, there exists some decomposition $0 = i_0 < \ldots < i_j = n$ such that for all natural numbers $1 \leq h \leq j$:

$$d_{edit}(w]i_{h-1}, i_h], L(A(Q_h))) < \frac{\varepsilon n}{2k}$$

Hence, we can correct all factors $w]i_{h-1}, i_h]$ into some word $w_h \in L(A(Q_h))$ at the cost of at most $\frac{\varepsilon n}{2k}$ edit operations. Let r_h be a successful run of $A(Q_h)$ on w_h. For $1 \leq h < j$, since $A(Q_h)$ is strongly connected and Π a component path of A, there is a word v_h of length at most $|Q|$ with a total quasi-run from $r_h(|w_h|)$ to $r_{h+1}(0)$. Since A is productive, there also exists a word v_0 of length at most $|Q|$ with a total run from *init* to $r_1(0)$, and another word v_j of length at most $|Q|$ with a total quasi-runs from $r_j(|v_j|)$ to *fin*. Now consider the word $w' = v_0 \cdot w_1 \cdot v_1 \cdots v_{j-1} \cdot w_j \cdot v_j$. Clearly, $w' \in L(A)$. Furthermore:

$$d_{edit}(w, w') \leq |Q| + \frac{\varepsilon n}{2k} \cdot j + (j-1)|Q| + |Q| = \frac{\varepsilon n j}{2k} + (j+1)|Q|$$

This inequality, $j \leq k$, and $n \geq \frac{2(k+1)|Q|}{\varepsilon}$ yield that $d_{edit}(w, L(A)) \leq \varepsilon n$. □

By combining Lemmas 11 and 7, we can show for all ε-far words, that there is a lower bound on the number of intervals blocking for every component path and decomposition along this path.

Lemma 12. *Let $\gamma' = 16k\delta(|Q|+1)$ (that is $\gamma' = 4k\gamma$). Then for any word $w \in \Sigma^n$ of length $n \geq 4\gamma'\lceil\log(\gamma')\rceil$ that is ε-far from $L(A)$, there exists a power of two l in $[2, \gamma']$ such that for all component paths $\Pi = (Q_1, \ldots, Q_j)$ of A and decompositions $0 = i_0 < \ldots < i_j = n$, there exists some interval $]i_{h-1}, i_h]$ of w that contains at least $n\beta'_l$ subintervals of size $2l$ are blocking for $A(Q_h)$, where $\beta'_l = l/(\gamma'\lceil\log(\gamma')\rceil)$.*

Proof. Let $w \in \Sigma^n$ be ε-far from $L(A)$ where $n \geq 4\gamma'\lceil\log(\gamma')\rceil$ and define $\beta'_l = l/(\gamma'\lceil\log(\gamma')\rceil)$ for all $l \in [2, \gamma']$. Note that $0 < \beta'_l \leq 1/4$. We consider an arbitrary component path $\Pi = (Q_1, \ldots, Q_j)$ of A. Since w is ϵ-far from $L(A)$, it follows with $\epsilon' = \epsilon/2k$ that w is also ϵ'-far

from Π by Lemma 11 (which can be applied since $n \geq 4\gamma'\lceil\log(\gamma')\rceil \geq \gamma' \geq \alpha$). Hence, for any decomposition $0 = i_0 < \ldots < i_j = n$ there exists an index h such that:

$$d_{edit}(w]i_{h-1}, i_h], L(A(Q_h))) > \frac{\varepsilon n}{2k}$$

Since $n \geq 4\gamma'\lceil\log(\gamma')\rceil$, we can apply Lemma 7 to the factor $w' = w]i_{h-1}, i_h]$, NFA $A' = A(Q_h)$, precision value ε', and γ' (instead of w, A, ϵ, and γ). Note that the size of w' is at most n as required. The lemma shows that there exists a power of two l' in $[2, \gamma']$, such that at least $n\beta'_{l'}$ intervals of w' blocking for $A(Q_h)$ of size $2l'$. Let l be the least l' for all component paths Π and decompositions $0 = i_0 < \ldots < i_j = n$. Since $\beta'_{l'}$ grows monotonically with l', the claim follows. □

Let w be a word of length n. For all natural numbers l, m, we define $\mathcal{S}(w, l, m)$ to be the set of all sequences containing m intervals of w of length l. A sequence $S = (I_1, \ldots, I_m)$ of $\mathcal{S}(w, l, m)$ is called blocking for A if and only if the fragment $F = \cup_{1 \leq o \leq m} I_o$ is blocking for A. Given a component path $\Pi = (Q_1, \ldots, Q_j)$, and a decomposition $J = (i_0, \ldots, i_j)$, we say that S matches (Π, J) if and only if there exists a non-blocking run of $A(\Pi)$ on $F = \cup_{1 \leq o \leq m} I_o$ such that $r(i) \in Q_h$ for $1 \leq h \leq j$ and all $i \in]i_{h-1}, i_h] \cap dom(F)$. In this case, we write $(\Pi, J) \models S$.

Lemma 13. *Let $\gamma' = 16k\delta(|Q| + 1)$. Then for any word $w \in \Sigma^n$ of length $n \geq 4\gamma'\lceil\log(\gamma')\rceil$ that is ε-far from $L(A)$ with respect to the edit distance, there exists a power of two l in $[2, \gamma']$ such that at least $\frac{2}{3}$ of the interval sequences in $\mathcal{S}(w, 2l, \alpha_l)$ are blocking for A, where $\alpha_l = 6k\gamma'\lceil\log(\gamma')\rceil^2/l$.*

Proof. Let $w \in \Sigma^n$ be ε-far from $L(A)$ where $n \geq 4\gamma'\log(\gamma')$. For an interval I of the domain of w, a set $Q' \subseteq Q$, and a natural number l, we denote by $\mathcal{B}(I, l, Q')$ the set of subintervals of size l of I that are blocking for $A(Q')$.

By Lemma 12, we can fix a power of two $l \in [2, \gamma']$, such that for all component paths $\Pi = (Q_1, \ldots, Q_j)$ of A and decompositions $J = (i_0, \ldots, i_j)$ for w and Π, there exists $1 \leq h \leq j$ such that the interval $I_h =]i_{h-1}, i_h]$ satisfies $|\mathcal{B}(I_h, 2l, Q_h)| \geq n\beta'_l$, where $\beta'_l = l/(\gamma'\lceil\log(\gamma')\rceil)$. In particular, note that the size of I_h is strictly greater than $n\beta'_l$ for such indexes h.

In order to obtain the lower bound in the lemma, we prove an upper bound on the number of nonblocking sequences in $\mathcal{S}(w, 2l, \alpha_l)$. The next claim shows that we can restrict ourselves to decompositions whose positions are multiples of $\lambda = n\beta'_l/4$, so that they belong to the set $\Lambda = \{\min(\lceil o\lambda\rceil, n) \mid 0 \leq o \leq 4/\beta'_l + 1\}$.

Claim 14. *For any nonblocking sequence $S \in \mathcal{S}(w, 2l, \alpha_l)$, there exist a strongly connected component Q_h of A and an interval $I =]i, i']$ with $i, i' \in \Lambda$ such that $\mathcal{B}(I, 2l, Q_h)$ contains at least 2λ intervals, but no interval of S.*

Let S be a nonblocking sequence in $\mathcal{S}(w, 2l, \alpha_l)$. By definition, there exists a component path $\Pi = (Q_1, \ldots, Q_j)$ and a decomposition $J = (i_0, \ldots, i_j)$ such that $(\Pi, J) \models S$. As argued above, there exists $1 \leq h \leq j$ such that the interval $I_h =]i_{h-1}, i_h]$ satisfies $|\mathcal{B}(I_h, 2l, Q_h)| \geq n\beta'_l = 4\lambda$. Since $(\Pi, J) \models S$, none of the intervals of S may belong to $\mathcal{B}(I_h, 2l, Q_h)$. Let I be the largest interval included in I_h with limits in Λ. By inclusion, no interval of sequence S may occur in $\mathcal{B}(I, 2l, Q_h)$ neither. Furthermore, the number of positions in I_h that are not in I is at most 2λ. Hence:

$$|\mathcal{B}(I, 2l, Q_h)| \geq |\mathcal{B}(I_h, 2l, Q_h)| - 2\lambda \geq 4\lambda - 2\lambda = 2\lambda$$

This concludes the proof of the claim.

For any interval I with limits in Λ and strongly connected component Q_h with $|\mathcal{B}(I, 2l, Q_h)| \geq 2\lambda$, the number of sequences in $\mathcal{S}(w, 2l, \alpha_l)$ without intervals from $\mathcal{B}(I, 2l, Q_h)$ is bounded by $(n - 2l - 2\lambda)^{\alpha_l}$. To obtain the lower bound on the total number of nonblocking sequences, we

```
fun membership_edit_A(ε, n, r_w)
  // r_w reference to an array containing some word w ∈ Σ^n
  // 0 < ε < 1 precision value
  // A = (Σ, Q, Δ, init, fin) NFA with init ≠ ∅ and fin ≠ ∅
  let k be the number of connected components of A
  let δ = 1/ε              // inverse precision
  let γ' = 16kδ(|Q| + 1) // size bound for small intervals
  if n < 4γ'⌈log(γ')⌉ then
     if w ∈ L(A) then return CLOSE else return NO
     // membership for small words w can be decided by running NFA A on w;
     // this needs time  O(γ'log(γ') |Q|)
  else // word w is sufficiently long
     for i = 1 to ⌈log(γ')⌉ do
       let l = min(2^i, γ')
       let α_l = 6kγ' log^2(γ')/l // number of intervals of size 2l
       for j = 1 to α_l
         select j ∈ [0, n − 2l] uniformly
         let I_j = ]j, j + 2l].
       end
       if fragment I_1 ∪ ... ∪ I_{α_l} of w is blocking wrt. A
          then return NO and exit else skip
     end
     return CLOSE
```

Figure 5: An approximate membership tester for NFAs with respect to the edit distance.

sum over all possible pairs of intervals and connected components. Their number is bounded by $|\Lambda|(|\Lambda|-1)k \leq 20k/{\beta'_l}^2$ so that the number of nonblocking sequences in $\mathcal{S}(w, 2l, \alpha_l)$ is at most $20k/{\beta'_l}^2 \cdot (n-2l-2\lambda)^{\alpha_l}$. For $k \geq 2$, the lemma now follows from the following estimation:

$$
\begin{array}{lcll}
\frac{20k}{{\beta'_l}^2} \cdot (n-2l-2\lambda)^{\alpha_l} & \leq & 20k/{\beta'_l}^2 \cdot (n-2l-n\frac{\beta'_l}{2})^{\alpha_l} & \text{since } \lambda = \frac{n\beta'_l}{4} \\
& \leq & 20k/{\beta'_l}^2 \cdot (n-2l)^{\alpha_l} \cdot (1-\frac{\beta'_l}{2})^{\alpha_l} & \text{since } \frac{n\beta'_l}{n-2l} \geq \beta'_l \\
& \leq & 20k/{\beta'_l}^2 \cdot |\mathcal{S}(w, 2l, \alpha_l)| \cdot (1-\frac{\beta'_l}{2})^{\alpha_l} & \text{since } |\mathcal{S}(w, 2l, \alpha_l)| = (n-2l)^{\alpha_l} \\
& \leq & 20k/{\beta'_l}^2 \cdot |\mathcal{S}(w, 2l, \alpha_l)| \cdot e^{\frac{-\alpha_l \beta'_l}{2}} & \\
& \leq & |\mathcal{S}(w, 2l, \alpha_l)| \cdot 20k/{\beta'_l}^2 \cdot e^{-3k\log(\gamma')} & \text{definition of } \alpha_l \text{ and } \beta'_l \\
& \leq & |\mathcal{S}(w, 2l, \alpha_l)| \cdot \frac{20k{\gamma'}^2\lceil\log^2(\gamma')\rceil}{2} \cdot (\frac{1}{\gamma'})^{3k} & \text{since } \beta'_l \geq \frac{2}{\gamma'\lceil\log(\gamma')\rceil} \\
& \leq & |\mathcal{S}(w, 2l, \alpha_l)| \cdot 10k \cdot (\frac{1}{\gamma'})^{3k-4} & \text{since } \log(\gamma') \leq \gamma' \\
& \leq & |\mathcal{S}(w, 2l, \alpha_l)| \cdot 10k \cdot (\frac{1}{\gamma'})^{2} & \text{since } k > 2 \\
& \leq & \frac{1}{3}|\mathcal{S}(w, 2l, \alpha_l)| & \text{since } \gamma' > 10k > 3
\end{array}
$$

The case $k = 1$ where A is strongly connected follows directly from Lemma 7. □

With this lemma and the fact that words from $L(A)$ have only feasible fragments, we obtain an approximated membership tester for NFAs which the expected performance.

Theorem 15. *Algorithm membership_edit$_A$ in Figure 5 is a one-sided approximate membership tester for words in languages defined by NFAs A with respect to the edit distance, with query complexity $O(\delta k^2|A|\log^3(\delta k|A|))$ and time complexity $O(\delta k^2|A|^2\log^3(\delta k|A|))$ where $\delta = 1/\epsilon$ is the inverse precision value and k the number of connected components in A.*

Proof. The case $n \leq 4\gamma'\lceil\log(\gamma')\rceil$ corresponds to an exact decision procedure so we can consider only the case $n > 4\gamma'\lceil\log(\gamma')\rceil$ in our argument. If $w \in L(A)$ then the path defined by any

successful run is matched by all sequences of intervals in w. Therefore the algorithm always answers CLOSE for such words. Now if w is ε-far from $L(A)$, then using Lemma 13 one has with probability at least $\frac{2}{3}$ a sequence $S \in \mathcal{S}(w, \alpha_l, 2l)$ which is blocking for A. Therefore the algorithm answers NO with at least the same probability. The query complexity is obtained by counting the accesses to $w]j, j+2l]$ for every selected j. Hence it is bounded by:

$$\sum_{i=1}^{\lceil \log(\gamma') \rceil} 2l\alpha_l \leq 8k\gamma' \lceil \log(\gamma') \rceil^3 = O(\delta k^2 |A| \log^3(\delta k |A|))$$

For the time complexity the consuming part corresponds to testing whether the sequences S is blocking, using the result of Proposition 3 we know that this can be done in time $O(|A|2l\alpha_l)$. Summing up yields a time complexity of

$$O(|A| \sum_{i=1}^{\lceil \log(\gamma') \rceil} 2l\alpha_l) = O(\delta k^2 |A|^2 \log^3(\delta k |A|))$$

□

6. Membership for NFAs modulo the Hamming Distance

We improve Alon et. al.'s approximate membership tester for DFAs [1] so that it runs in polynomial time, while being extended to NFAs. The correctness argument follows the same schema as worked out there for the original algorithm. Therefore, we present only a sketch of the proof in which we point out the differences to before.

The main difference of our improved algorithm is that we rely on deciding feasibility of fragments. The original algorithm decided infeasibility for intervals in the case of DFAs with single strongly connected components. In the general case, it relied on deciding infeasibility of fragments only implicitly, without having extracted that notion. It should also be noticed that we elaborated the same schema again for our new tester for NFAs with respect to the edit distance, with the main difference that infeasible fragments are used there instead of blocking fragments.

6.1. Strongly Connected NFAs

We start with strongly connected NFAs A. For any word that is far from the language of A with respect to the Hamming distance, we show that it has many small infeasible intervals. This is stated by the following lemma, which is the analogous of our Lemma 7 in the case of the Hamming distance, except that it was only stated for DFAs in [1].

Lemma 16 (Lemma 2.4 of [1]). *Let A be a strongly connected* NFA *over Σ and $\delta = 1/\varepsilon$. Then there exists a natural number $m \leq 3|Q|^2$ (called the reachability constant of A) such that for any word $w \in \Sigma^n$ of length $n \geq 64\delta m \log(4m\delta)$ such that $d_{hamming}(w, L(A)) \geq \varepsilon n$, there exists a power of two $l \in [2, 4m\delta]$ such that the number of infeasible intervals of w of length $2l$ is at least $2ln/(\delta m \log(4m\delta))$.*

The proof is analogous to the proof of our Lemma 7 for the edit distance. With the Hamming distance, however, one must adapt the repair strategy carefully, such that it produces a word of the exactly the same size n. The fact that we lift this lemma from DFAs to NFAs does not matter, since the original proof did not depend on determinism.

According to Proposition 5, we can decide feasibility of intervals in polynomial time in the size of the interval and the NFA, and independently of the size of the word. In combination with Lemma 16 this allow us to construct an approximate membership tester with constant query complexity (in the size of the word) that runs in polynomial time for all NFAs that are strongly connected.

6.2. General NFAs

In the case of NFAs with many strongly connected components, we consider components path with decomposition of the word, as we did before for the edit distance. Let $A = (\Sigma, Q, \Delta, init, fin)$ be an NFA with k strongly connected components. Without loss of generality, we assume that A is productive. Let $0 < \varepsilon < 1$ and $\delta = 1/\varepsilon$. For any state set Q' we denote by $A(Q', p, q)$ the NFA $(\Sigma, Q', \Delta', \{p\}, \{q\})$ where Δ' is the restriction of Δ to states in Q'. Let $w \in \Sigma^*$. A triplet $(\Pi, J, (p_o, q_o)_{1 \le o \le j})$ consists of a component path $\Pi = (Q_1, \ldots, Q_j)$, a decomposition $J = (i_0, \ldots, i_j)$ of w, where $p_o, q_o \in Q_o$ for all $1 \le o \le j$. A triplet is called admissible if $init \to_A p_0$, $q_j \to_A fin$ and and for all $1 \le o < j$ it holds that $q_o \to_A p_{o+1}$ and $p_o \to_A^{i_{o+1}-i_o-1} q_o$. The next lemma relates the general case to the case with one strongly connected component.

Lemma 17 (Lemma 2.7 in [1]). *Let $(\Pi, J, (p_o, q_o)_{1 \le o \le j})$ be an admissible triplet for a component path $\Pi = (Q_1, \ldots, Q_j)$ and a decomposition $J = (i_0, \ldots, i_j)$ of a word w. It w is ϵ-far from $L(A)$ with respect to the Hamming distance, then there exists $1 \le h \le j$ such that:*

$$d_{hamming}(w]i_{h-1}, i_h[, L(A(Q_h, p_h, q_h))) \ge \frac{\epsilon|w|}{2k}$$

The proof of this lemma is similar to the proof of Lemma 11, except that the repair strategy needs to be adapted so that it copes with Hamming distance properly, as done before for Alon et. al.'s algorithm. We omit the details.

Lemma 18. *Let M be the maximal reachability constants m for all connected components in A (so $M = 3|A|^2$ in the worst case) and $\eta = 8Mk\delta$. For any word $w \in \Sigma^n$ of length $n \ge 16\eta \log(\eta)$ that is ε-far from $L(A)$ with respect to the Hamming distance, there exists a power of two l in $[2, \eta]$ such that at least $\frac{2}{3}$ of the elements in $\mathcal{S}(w, 2l, \frac{3k\eta \log^2(\eta)}{4l})$ are infeasible for A.*

This lemma can be proven in the same way than Lemma 13. The only thing that changes is that blocking intervals or fragments are to be exchanged for infeasible intervals or fragments.

Theorem 19. *Algorithm membership_hamming$_A$ in Figure 6 is a one-sided approximate membership tester for NFAs A with respect to the Hamming distance. If k the number of strongly connected components of A and $\delta = 1/\epsilon$ the inverse precision, then its query complexity is in $O(\delta k^2 |A|^2 \log^3(\delta k |A|^2))$ and its time complexity in $O(\delta k^2 |A|^5 \log^3(\delta k |A|^2))$.*

Proof. Based on Lemma 18, we can argue as we did for the edit distance, that algorithm *membership_hamming$_A$* is a membership tester with respect to the Hamming distance. Its query complexity and running time are mainly due to deciding the feasibility of the randomly selected sequence of intervals. By Proposition 5, we can compute the feasibility of a fragment F in time $O(|F||A|^3)$ after a global precomputation (once for all fragments) in $O(|A|^5)$. □

7. Conclusion and Future Work

We have shown that approximate membership testing for DFAs with constant query complexity can be done in polynomial time. It turns out that approximation modulo the edit distance leads to more natural algorithms with lower query and time complexity.

This opens up quite some questions for future work. First of all, we would like to develop approximate membership testing algorithms for regular tree languages. We conjecture that this will not be possible without imposing serious locality restrictions, but this needs to be elaborated. Since document type descriptors (DTDs) for XML documents satisfy strong locality restrictions,

```
fun membership_hamming_A(ε, n, r_w)
  // r_w reference to an array containing some word w ∈ Σ^n
  // 0 < ε < 1 precision value
  // A = (Σ, Q, Δ, init, fin) NFA with init ≠ ∅ and fin ≠ ∅.
  let k be the number of connected components of A
  let M be the maximal reachability constant m of all strongly connected
      components of A
      // M can be computed in time O(|A|^2) as shown in the original algorithm
      // or else we can choose the worst case M = 3|Q|^2
  let δ = 1/ε // inverse error precision
  let η = 8Mkδ
  if n < 16η log(η) then
     if w ∈ L(A) then return CLOSE else return NO
     // membership for small words w can be decided by running NFA A on w;
     // this needs time O(ηlog(η) |Q|)
  else // word w is sufficiently long
     for i = 1 to ⌈log(η)⌉ do
       let l = min(2^i, η)
       for j = 1 to (3kη log^2(4η)) / l
         select j ∈ [0, n − 2l] from a uniform distribution
         let I_j = ]j, j + 2l]
       end
       if fragment I_1 ∪ ... ∪ I_{α_l} of w is infeasible wrt. A
          then return NO and exit else skip
     end
     return CLOSE
```

Figure 6: An approximate membership tester for NFAs with respect to the Hamming distance.

one could hope for approximate membership testers for (DTDS) and thus for efficient approximate XML schema validation. First results in this direction exist already for the edit distance with moves [14], but one could hope for better results for the edit distance in the future.

Another direction would be to study approximate inclusion or equivalence checking with respect to the edit distance [3]. So far, approximate inclusion checking has only been considered for the edit distance with moves [6]. So the question is whether approximation can lead to realistic algorithms in this perspective.

Acknowledgements

This work is part of the project ENUM of the program ANR BLANC, so we thank all members of this project. In particular, we would like to thank Arnaud Durand, the coordinator of ENUM, for having brought the authors together. He also introduced us to Michel De Rougemont, who was the main source of our growing interest in property testing. We would like to thank Michel, who contributed his expertise through many helpful discussions. Benoit Groz and Iovka Boneva from the Mostrare project, who pointed out to us the Chrobak normal form.

References

[1] Noga Alon, Michael Krivelevich, Ilan Newman, and Mario Szegedy. Regular languages are testable with a constant number of queries. *Society for Industrial and Applied Mathematics Journal on Computing*, 30(6):1842–1862, 2000.

[2] Noga Alon and Asaf Shapira. Testing satisfiability. In *Symposium on Discrete Algorithms*, pages 645–654, 2002.

[3] Michael Benedikt, Gabriele Puppis and Cristian Riveros. The Cost of Traveling between Languages. In *the 38th International Colloquium on Automata, Languages and Programming*, volume 6756 of *Lecture Notes in Computer Science*, pages 234–245. Springer, 2011.

[4] Artur Czumaj and Christian Sohler. Abstract Combinatorial Programs and Efficient Property Testers. In *Society for Industrial and Applied Mathematics Journal on Computing*, 34(3): 580–615, 2005.

[5] Eldar Fischer. The art of uninformed decisions. *Bulletin of the European Association of Theoritical Computer Science*, 75:97, 2001.

[6] Eldar Fischer, Frederic Magniez, and Michel de Rougemont. Approximate satisfiability and equivalence. In *Society for Industrial and Applied Mathematics Journal on Computing*, 39(6): 2251–2281, 2010.

[7] Pawel Gawrychowski. Chrobak normal form revisited, with applications. In *the International Conference on Implementation and Application of Automata*, volume 6807 of *Lecture Notes in Computer Science*, pages 142–153. Springer, 2011.

[8] Oded Goldreich. Combinatorial property testing (a survey). In *Randomization Methods in Algorithm Design*, pages 45–60, 1998.

[9] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the Association for Computing Machinery*, 45(4):653–750, 1998.

[10] Ilan Newman and Christian Sohler. Every property of hyperfinite graphs is testable. In *Symposium on Theory of Computing*, pages 675–684. Association for Computing Machinery, 2011.

[11] Ilan Newman. Testing membership in languages that have small width branching programs. In *Society for Industrial and Applied Mathematics Journal on Computing*, 3142(5):1557–1570, 2002.

[12] Dana Ron. Property Testing: A Learning Theory Perspective. In *Foundations and Trends in Machine Learning*, 1(3): 307–402, 2008.

[13] Dana Ron, Ronitt Rubinfeld, Muli Safra and Omri Weinstein. Approximating the Influence of Monotone Boolean Functions in $O(\sqrt{n})$ Query Complexity In Random Approx, 664–675, 2011.

[14] Michel de Rougemont and Adrien Vieilleribiere. Approximate Structural Consistency. *SOFtware SEMinar*, pages 685-696, 2010.

[15] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *Society for Industrial and Applied Mathematics Journal on Computing*, 25(2):252–271, 1996.